Name : Nekkanti Uday Dinesh

Property price Indian cities dataset

About Dataset

This dataset was scraped from one of the housing website called as makaan.com. Makaan.com has quickly emerged as the preferred partner for consumers looking to rent, buy or sell a home. Makaan.com offers its online consumers maximum property options and has become one of the largest advertising platforms in online real estate in India.

Some Key Details

Property_Name :Property Holder Name
Property_id :Property Holder Unique ID number
Property_type :Types of Property
Property_status :Status of Property
Price_per_unit_area :Price per unit area in rupees
Posted_On :It's about when they posted poject on website
Project_URL :The link of the project
builder_id :Builder Unique ID number
Builder_name :Builder Name
Property_building_status :About building status Active or Inactive
City_id :City ID number
City_name :City name
No_of_BHK :Number of BHK
Locality_ID :Locality ID number
Locality_Name :Locality name
Longitude :Longitude of area
Latitude :Latitude of area
Price :Price of property
Size :Size of property
Sub_urban_ID :Sub urban ID number
Sub_urban_name :Sub urban name
description :Description about property
is_furnished :How many are furnished
listing_domain_score :Score of domain
is_plot : Whether it is plotted or not
is_RERA_registered :RERA Registered
is_Apartment :Whether it is apartment or not
is_ready_to_move :How many are ready to move

is_commercial_Listing :The property is commercial or not.
is_PentaHouse :The property is pent house or not
is_studio :The property is studio or not
Listing_Category :Whether the property is available or not

What insights do I hope to glean from this data?

• How to read the flies , last five rows , first five rows ? Describe, size and information of dataset?

• Identifying the null values and removing with certain methods.

• How many properties are there in each city?

• What are the Top 25 No. of Properties by Price per unit area ?

• How many properties are there in between certain square feets?

• How much percentage of apartments are there in given dataset?

• How many properties are there in each BHK?

• Domain Score of properties in given dataset.

• What type of properties data set have ?

• What are properties status in each city ?

• How many are furnished and RERA registered in each city?

• Comparing Price of properties in each city?

• Comparing prices of apartments and BHKs in each city?

• Top 25 locality areas with locations.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import pyplot
import seaborn as sns
import plotly
import plotly.express as px
import plotly.graph_objects as go
import missingno as msno
import warnings
warnings.filterwarnings('ignore')
```

• There are various other methods to read different types of files, such as read_csv(), read_json(), read_html(), read_excel(), etc which can be easily used as per the requirement.

• head() returns the first n rows(observe the index values). The default number of elements to display is five, but you may pass a custom number.

In [2]:

```
df = pd.read_csv('Makaan_Properties_Buy.csv')
df.head()
```

Out[2]:

	Property_Name	Property_id	Property_type	Property_status	Price_per_unit_area	Posted_On
0	Arkiton Luxe	15446514	Apartment	Under Construction	4,285	1 day ago
1	Keshav Akshar Ocean Pearl	15367414	Apartment	Under Construction	7,000	2 days ago
2	Vishwa Opulence	14683118	Apartment	Ready to move	5,752	2 days ago
3	Satyam Sarjan	5476295	Apartment	Ready to move	2,486	5 days ago
4	Navkar Sunflower	15477040	Apartment	Under Construction	5,324	8 days ago

5 rows × 32 columns

• tail() returns the last n rows(observe the index values). The default number of elements to display is five, but you may pass a custom number.

In [3]:

```
df.tail()
```

	Property_Name	Property_id	Property_type	Property_status	Price_per_unit_area	Posted_On	
332091	RajlaxmiRaajLaxmiTowers	10324765	Apartment	Under Construction	9,826	4 months ago	https://
332092	RajlaxmiRaajLaxmiTowers	15076701	Apartment	Under Construction	8,568	4 months ago	https://
332093	RajlaxmiRaajLaxmiTowers	10324762	Apartment	Under Construction	9,861	4 months ago	https://
332094	RajlaxmiRaajLaxmiTowers	15076700	Apartment	Under Construction	8,813	4 months ago	https://
332095	RajlaxmiRaajLaxmiTowers	14683740	Apartment	Under Construction	9,859	4 months ago	https://

• To get the shape of Pandas DataFrame, use ‘df. Shape’. The shape property returns a tuple representing the dimensionality of the DataFrame. The format of shape would be (rows, columns).

In [4]:

```
df.shape
```

Out[4]:

```
(332096, 32)
```

• ‘df.info()’.The info() method prints information about the DataFrame. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values). Note: the info() method actually prints the info.

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 332096 entries, 0 to 332095
Data columns (total 32 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   Property_Name             217826 non-null  object 
 1   Property_id               332096 non-null  int64  
 2   Property_type             332096 non-null  object 
 3   Property_status           271654 non-null  object 
 4   Price_per_unit_area       332096 non-null  object 
 5   Posted_On                 332096 non-null  object 
 6   Project_URL               332096 non-null  object 
 7   builder_id                149978 non-null  float64
 8   Builder_name              149978 non-null  object 
 9   Property_building_status  332096 non-null  object 
 10  City_id                   332096 non-null  int64  
 11  City_name                 332096 non-null  object 
 12  No_of_BHK                 332096 non-null  object 
 13  Locality_ID               332096 non-null  int64  
 14  Locality_Name             332094 non-null  object 
 15  Longitude                 332096 non-null  float64
 16  Latitude                  332096 non-null  float64
 17  Price                     332096 non-null  object 
 18  Size                      332096 non-null  object 
 19  Sub_urban_ID              332096 non-null  int64  
 20  Sub_urban_name            332096 non-null  object 
 21  description               332095 non-null  object 
 22  is_furnished              332096 non-null  object 
 23  listing_domain_score      332096 non-null  float64
 24  is_plot                   332096 non-null  bool   
 25  is_RERA_registered        332096 non-null  bool   
 26  is_Apartment              332096 non-null  bool   
 27  is_ready_to_move          332096 non-null  bool   
 28  is_commercial_Listing     332096 non-null  bool   
 29  is_PentaHouse             332096 non-null  bool   
 30  is_studio                 332096 non-null  bool   
 31  Listing_Category          332096 non-null  object 
dtypes: bool(7), float64(4), int64(4), object(17)
memory usage: 65.6+ MB
```

Coverting Data Types

Python defines type conversion functions to directly convert one data type to another which is useful to do Statistical analysis. Some are there in string datatype I have converted into float datatype.

The Cloumns I have changed(Object to Float) are : No_of_BHK , Price , Price_per_unit_area , Size.

In [6]:

```
df['No_of_BHK'] = df['No_of_BHK'].apply(lambda x: float(x.split()[0].replace('BHK', '')))
df['Price'] = df['Price'].apply(lambda x: float(x.split()[0].replace(',', '')))
df['Price_per_unit_area'] = df['Price_per_unit_area'].apply(lambda x: float(x.split()[0].re
```

In [7]:

```
df['Size'] = df['Size'].apply(lambda x: str(x.split()[0].replace(',','')))
df['Size'] = df['Size'].apply(lambda x: str(x.split()[0].replace('sq ft','')))
df['Size'] = df['Size'].astype(float)
```

After Converting Data Types

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 332096 entries, 0 to 332095
Data columns (total 32 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   Property_Name             217826 non-null  object
 1   Property_id               332096 non-null  int64
 2   Property_type             332096 non-null  object
 3   Property_status           271654 non-null  object
 4   Price_per_unit_area       332096 non-null  float64
 5   Posted_On                 332096 non-null  object
 6   Project_URL               332096 non-null  object
 7   builder_id                149978 non-null  float64
 8   Builder_name              149978 non-null  object
 9   Property_building_status  332096 non-null  object
 10  City_id                   332096 non-null  int64
 11  City_name                 332096 non-null  object
 12  No_of_BHK                 332096 non-null  float64
 13  Locality_ID               332096 non-null  int64
```

• The describe() method returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains these information for each column: count - The number of not-empty values. mean - The average (mean) value.This includes count, mean, median (or 50th percentile) standard variation, min-max, and percentile values of columns.

Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding `NaN` values.

Analyzes both numeric and object series, as well as `DataFrame` column sets of mixed data types. The output will vary depending on what is provided.

In [9]:

```
df.describe()
```

Out[9]:

	Property_id	Price_per_unit_area	builder_id	City_id	No_of_BHK	Loca
count	3.320960e+05	332096.000000	1.499780e+05	332096.000000	332096.000000	332096.(
mean	1.319382e+07	8650.011072	1.108900e+07	12.137861	1.733357	63082.9
std	2.533792e+06	10902.350399	3.100384e+07	7.270491	1.407407	26246.9
min	5.000114e+06	0.000000	1.000020e+05	1.000000	0.000000	50001.(
25%	1.244466e+07	2999.000000	1.006780e+05	5.000000	0.000000	50378.(
50%	1.419741e+07	5571.000000	1.034750e+05	12.000000	2.000000	51893.(
75%	1.509555e+07	10114.250000	6.547740e+05	18.000000	3.000000	60223.(
max	1.558147e+07	200000.000000	1.007295e+08	23.000000	15.000000	173237.(

• You will see the percentiles(25%, 50%, 75%..etc) and some values in front of them. The significance is to tell you the distribution of your data. 25% means 25% of your data have the value 1.0000 or below.

Data Cleaning

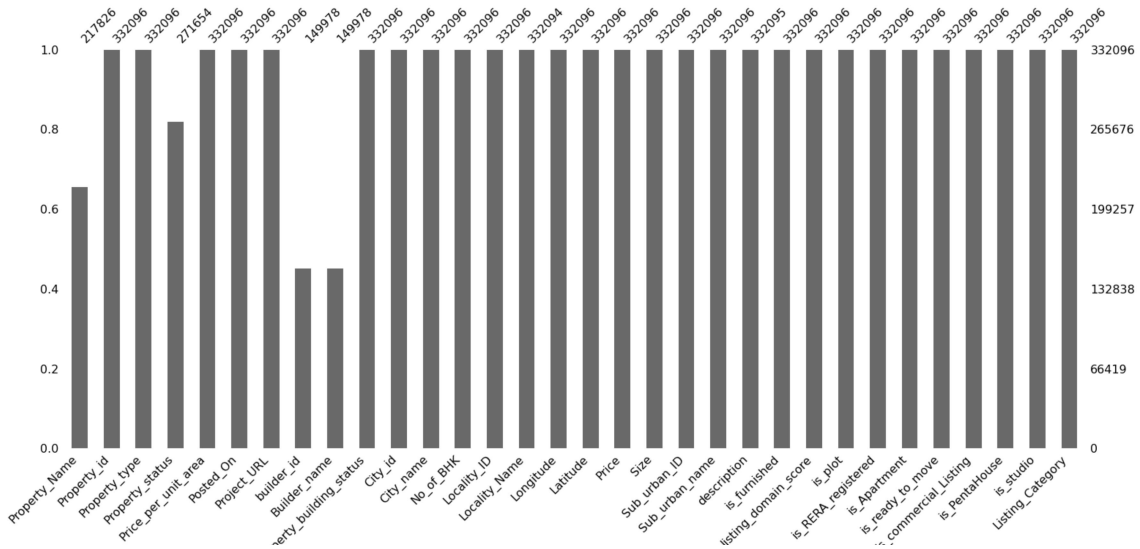
• I have used Missingno library. It is simple to use Python library that provides a series of visualisations to understand the presence and distribution of missing data within a pandas dataframe. This can be in the form of either a barplot, matrix plot, heatmap, or a dendrogram. I have represented bar graph. This shows existing values.

In [10]:

```
msno.bar(df)
```

Out[10]:

<AxesSubplot:>



•'isnull(). sum()'.It will return the count of null values in each column.It will count only the standard null values.Since sum() calculate as True=1 and False=0 , you can count the number of missing values in each row and column by calling sum() from the result of isnull() . You can count missing values in each column by default, and in each row with axis=0.

In [11]:

```
df.isnull().sum()
```

Out[11]:

```
Property_Name               114270
Property_id                      0
Property_type                    0
Property_status              60442
Price_per_unit_area              0
Posted_On                        0
Project_URL                      0
builder_id                  182118
Builder_name                182118
Property_building_status         0
City_id                          0
City_name                        0
No_of_BHK                        0
Locality_ID                      0
Locality_Name                    2
Longitude                        0
Latitude                         0
Price                            0
Size                             0
Sub_urban_ID                     0
Sub_urban_name                   0
description                      1
is_furnished                     0
listing_domain_score             0
is_plot                          0
is_RERA_registered               0
is_Apartment                     0
is_ready_to_move                 0
is_commercial_Listing            0
is_PentaHouse                    0
is_studio                        0
Listing_Category                 0
dtype: int64
```

Droping value

In [12]:

```
df['Locality_Name'].value_counts()
```

Out[12]:

```
Thane West                      10676
Mira Road East                  10315
Sultanpur Road                   4388
Dombivali                        3491
New Town                         3123
                                ...
RP Road                             1
Singareni Colony                    1
Keesara Hyderabad                   1
Himayatnagar                        1
Samata Nagar Thakur Village         1
Name: Locality_Name, Length: 5046, dtype: int64
```

• I have used dropna() for Locality_Name cloumns sinces there are 2 null values. The dropna() method removes the rows that contains NULL values. The dropna() method returns a new DataFrame object unless the inplace parameter is set to True , in that case the dropna() method does the removing in the original DataFrame instead.

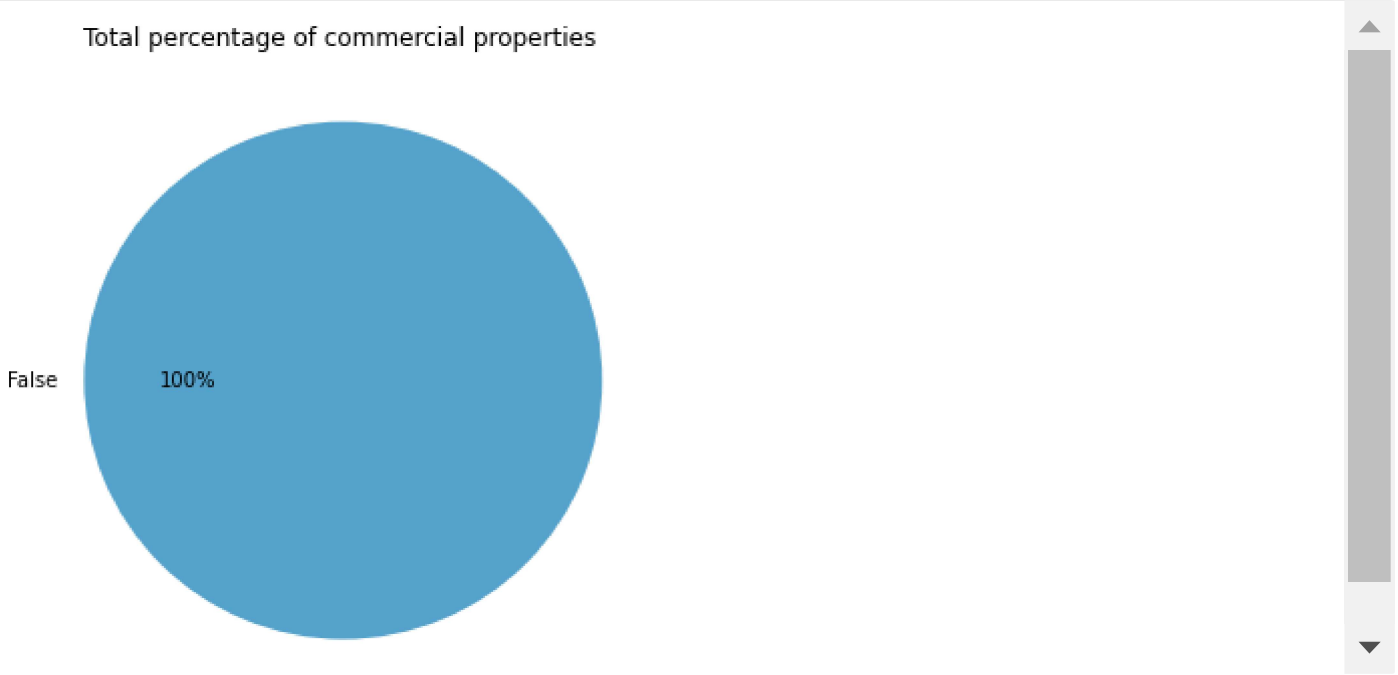
In [13]:

```
df.dropna(subset=['Locality_Name'],inplace = True )
```

In is_commercial_Listing column all are false so directly we can remove it as it i
s not useful in further analysis.

In [14]:

```
plt.figure(figsize = (10, 5))
data = df['is_commercial_Listing'].value_counts()
colors = sns.color_palette('icefire')
labels = ['False']
plt.pie(data, labels = labels, autopct = '%.0f%%', colors = colors)
plt.title('Total percentage of commercial properties ')
plt.tight_layout()
```



The drop() method removes the specified row or column. By specifying the column axis (axis='columns'), the drop() method removes the specified column. By specifying the row axis (axis='index'), the drop() method removes the specified row. inplace=True means the operation would work on the original object. axis=1 means we are dropping the column, not the row.

In [15]:

```
df=df.drop('is_commercial_Listing', axis=1)
```

Filling with NAN

• I have filled with NAN because some columns have huge null values and they are important columns so I have replaced with NAN for further data analysis.

The fillna() function is used to fill NA/NaN values using the specified method. Value to use to fill holes, alternately a dict/Series/DataFrame of values specifying which value to use for each index or column.

In [16]:

```
df1 = df[['Property_Name','Property_status']] = df[['Property_Name','Property_status']].fil
print(df1)
```

```
                   Property_Name     Property_status
0                    Arkiton Luxe  Under Construction
1       Keshav Akshar Ocean Pearl  Under Construction
2                  Vishwa Opulence       Ready to move
3                    Satyam Sarjan       Ready to move
4                 Navkar Sunflower  Under Construction
...                            ...                ...
332091  Rajlaxmi RaajLaxmi Towers  Under Construction
332092  Rajlaxmi RaajLaxmi Towers  Under Construction
332093  Rajlaxmi RaajLaxmi Towers  Under Construction
332094  Rajlaxmi RaajLaxmi Towers  Under Construction
332095  Rajlaxmi RaajLaxmi Towers  Under Construction

[332094 rows x 2 columns]
```

In [17]:

```
df2 = df[['builder_id','Builder_name' ]] = df[['builder_id','Builder_name' ]].fillna('NAN')
print(df2)
```

```
         builder_id                 Builder_name
0       100563465.0           Arkiton life Space
1       100009433.0         Keshav Narayan Group
2       100207731.0  Vishwa Developers Ahmedabad
3          101303.0            Satyam Developers
4         1484209.0    Navkar Buildcon Ahmedabad
...             ...                          ...
332091     561277.0   Rajlaxmi Developers Mumbai
332092     561277.0   Rajlaxmi Developers Mumbai
332093     561277.0   Rajlaxmi Developers Mumbai
332094     561277.0   Rajlaxmi Developers Mumbai
332095     561277.0   Rajlaxmi Developers Mumbai

[332094 rows x 2 columns]
```

In [18]:

```
df3= df['description']=df['description'].fillna('Not mentioned')
print(df3)
```

```
0         The house is unfurnished. It has car parking. ...
1         A 4 bhk property is available for sale in Kesh...
2         It has a built-up area of 2295 sqft and is pri...
3         It’s a 2 bhk multistorey apartment situated in...
4         A 3 bhk property is available for sale in Navk...
                              ...
332091    It’s a 1 bhk multistorey apartment situated in...
332092    It has an area of 426 sqft . The property is a...
332093    Well designed 1 bhk multistorey apartment is a...
332094    It’s a 2 bhk multistorey apartment situated in...
332095    A spacious 1 bhk multistorey apartment is avai...
Name: description, Length: 332094, dtype: object
```

In [19]:

```
df.isnull().sum()
```

Out[19]:

```
Property_Name               0
Property_id                 0
Property_type               0
Property_status             0
Price_per_unit_area         0
Posted_On                   0
Project_URL                 0
builder_id                  0
Builder_name                0
Property_building_status    0
City_id                     0
City_name                   0
No_of_BHK                   0
Locality_ID                 0
Locality_Name               0
Longitude                   0
Latitude                    0
Price                       0
Size                        0
Sub_urban_ID                0
Sub_urban_name              0
description                 0
is_furnished                0
listing_domain_score        0
is_plot                     0
is_RERA_registered          0
is_Apartment                0
is_ready_to_move            0
is_PentaHouse               0
is_studio                   0
Listing_Category            0
dtype: int64
```

Removing Outliers

I have used plotly for better user experience and visualization.

• I have removed outliers by using Box Plot. Box plots are used to show distributions of numeric data values, especially when you want to compare them between multiple groups. They are built to provide high-level information at a glance, offering general information about a group of data's symmetry, skew, variance, and outliers.

In [20]:

```
px.box(df['No_of_BHK'])
```

• For removing Outilers , I have used IQR method. We can use the IQR method of identifying outliers to set up a “fence” outside of Q1 and Q3. Any values that fall outside of this fence are considered outliers. To build this fence we take 1.5 times the IQR and then subtract this value from Q1 and add this value to Q3.

In [21]:

```
Q1 = np.percentile(df["No_of_BHK"], 25, interpolation = "midpoint")
Q3 = np.percentile(df["No_of_BHK"], 75, interpolation = "midpoint")
IQR = Q3 - Q1
min1 = Q1 - 1.5 * IQR
max1 = Q3 + 1.5 * IQR
print(min1, max1, IQR, Q1, Q3)
```

```
-4.5 7.5 3.0 0.0 3.0
```

In [22]:

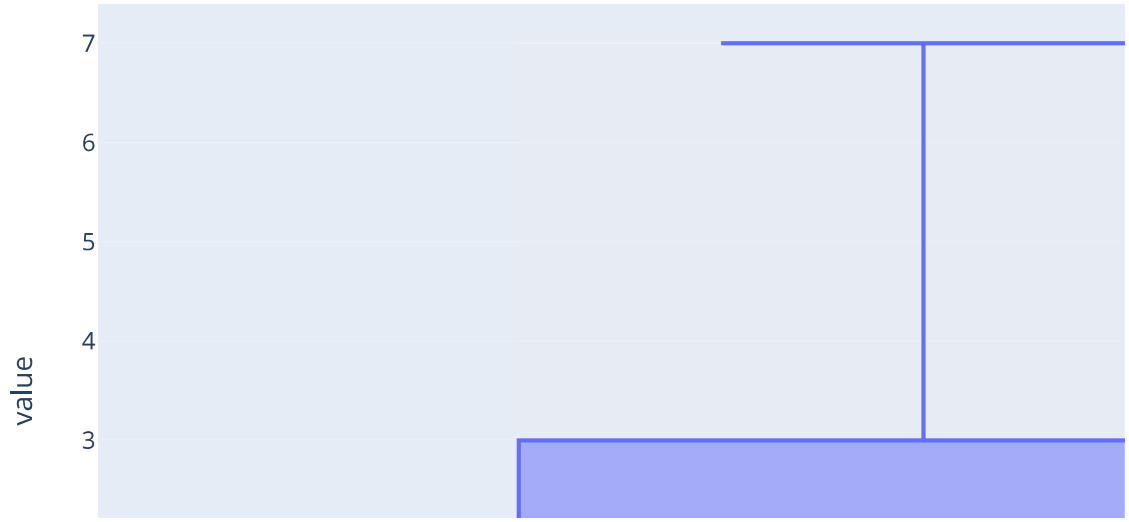
```
df = df[(df["No_of_BHK"] > min1) & (df["No_of_BHK"] < max1)]
df.shape
```

Out[22]:

```
(331305, 31)
```

In [23]:

```
px.box(df['No_of_BHK'])
```



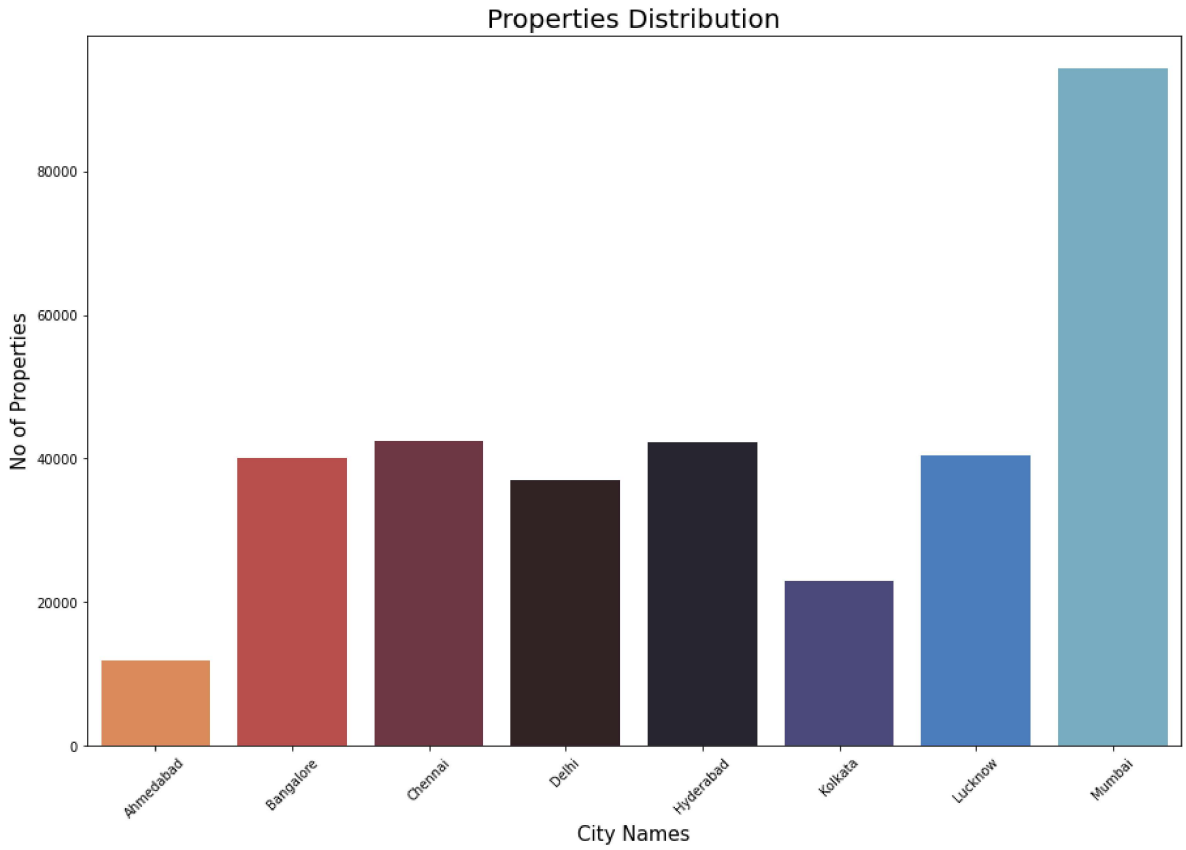
Univariate, Bivariate and Multivariate analysis with graphs and Explanations

Univariate Analysis

The countplot is majorly used for showing the observational count in different category based bins with the help of bars. In this I have showed properties distribution in each city.

In [24]:

```
fig, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Properties Distribution', fontsize=20)
sns.countplot(x='City_name',data=df,palette='icefire_r')
ax.set_ylabel('No of Properties', fontsize=15)
ax.set_xlabel('City Names', fontsize=15)
plt.xticks(rotation=45)
plt.show()
```



In [25]:

```
df['City_name'].value_counts()
```

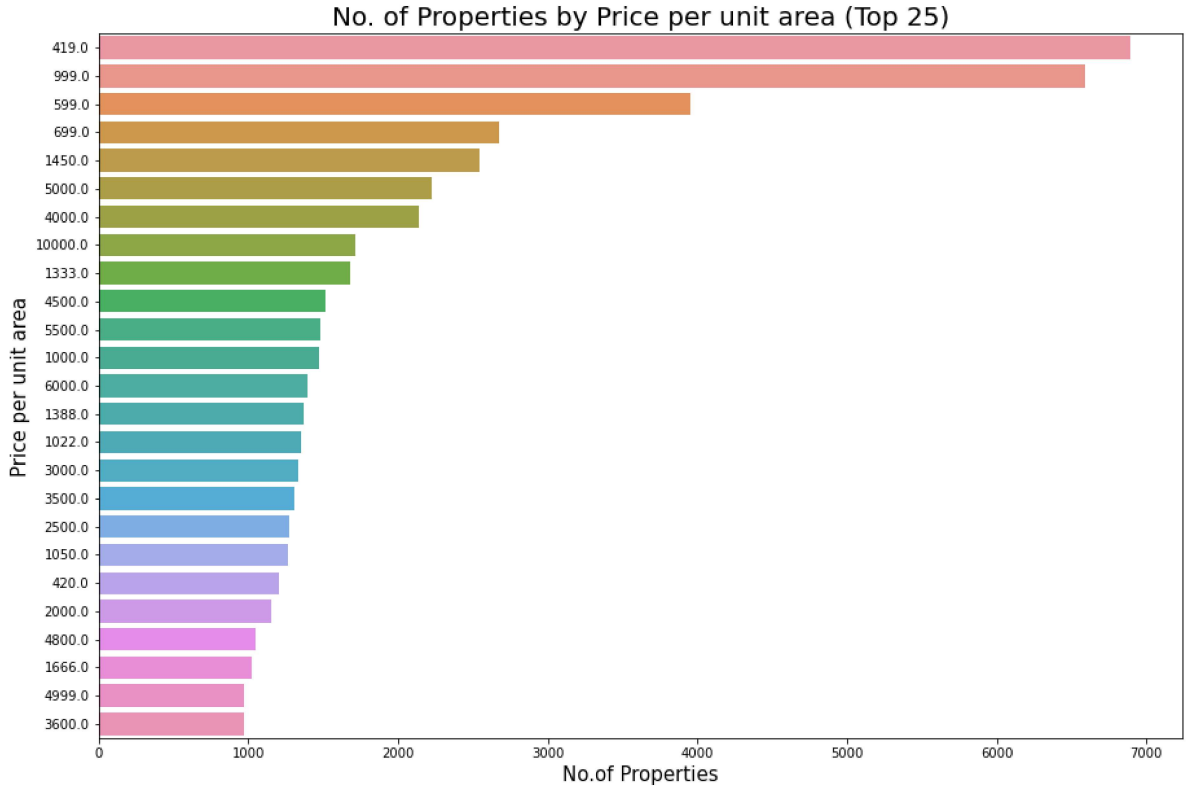
Out[25]:

```
Mumbai       94272
Chennai      42379
Hyderabad    42231
Lucknow      40369
Bangalore    40158
Delhi        37064
Kolkata      22900
Ahmedabad    11932
Name: City_name, dtype: int64
```

• I have used Countplot for No. of Properties by Price per unit area. I have taken TOP 25 values So, People will get know how many properties are there in certain price. It will make easy to buy properties

In [26]:

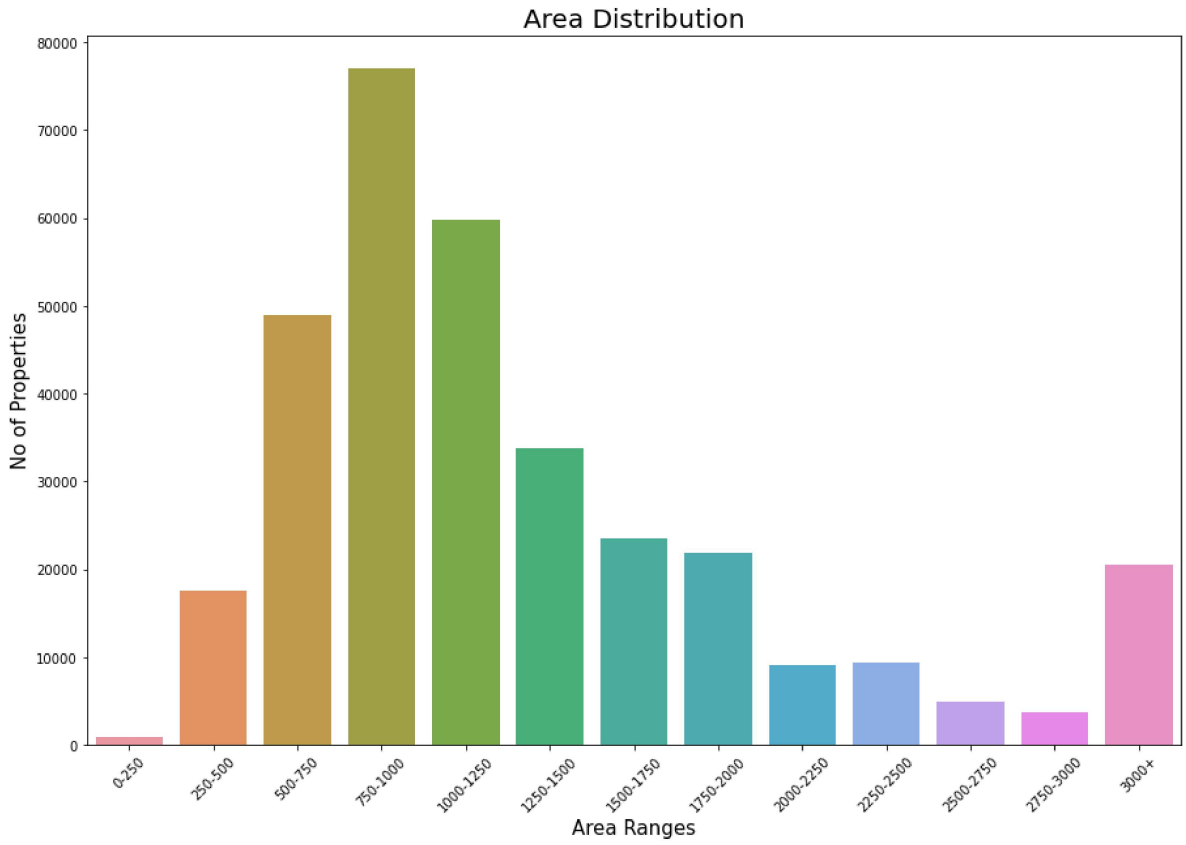
```
fig, ax = plt.subplots(figsize=(15, 10))
ax.set_title('No. of Properties by Price per unit area (Top 25)', fontsize=20)
sns.countplot(y='Price_per_unit_area', data=df, order=df.Price_per_unit_area.value_counts()
ax.set_xlabel('No.of Properties', fontsize=15)
ax.set_ylabel('Price per unit area', fontsize=15)
plt.show()
```



• I have done distribution of area of properties,in this we can see between 750-1000 sq.ft have more no.of properties. it will be easy to analysis and visualization that how many properties are there in certain area.

In [27]:

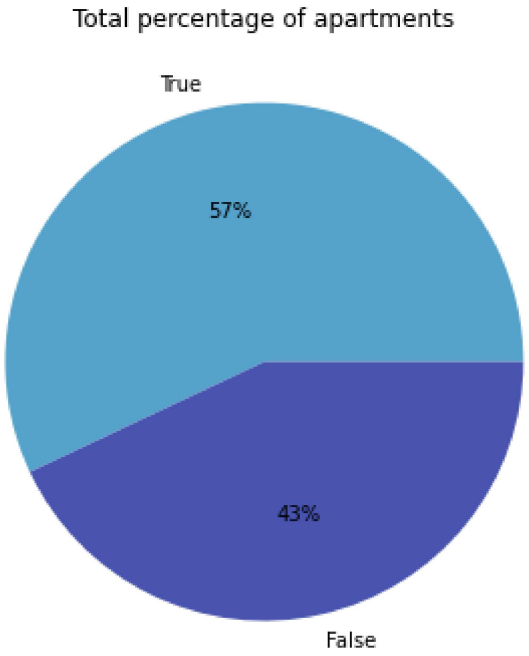
```
df['Size'] = pd.cut(df['Size'], bins=[0, 250, 500, 750, 1000, 1250, 1500, 1750, 2000, 2250,
                    labels=['0-250', '250-500', '500-750', '750-1000', '1000-1250', '1250-1
fig, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Area Distribution', fontsize=20)
sns.countplot(x='Size', data=df)
ax.set_ylabel('No of Properties', fontsize=15)
ax.set_xlabel('Area Ranges', fontsize=15)
plt.xticks(rotation=45)
plt.show()
```



I have used Pie chart to show how much percentage of apartments are there in my dataset.

In [28]:

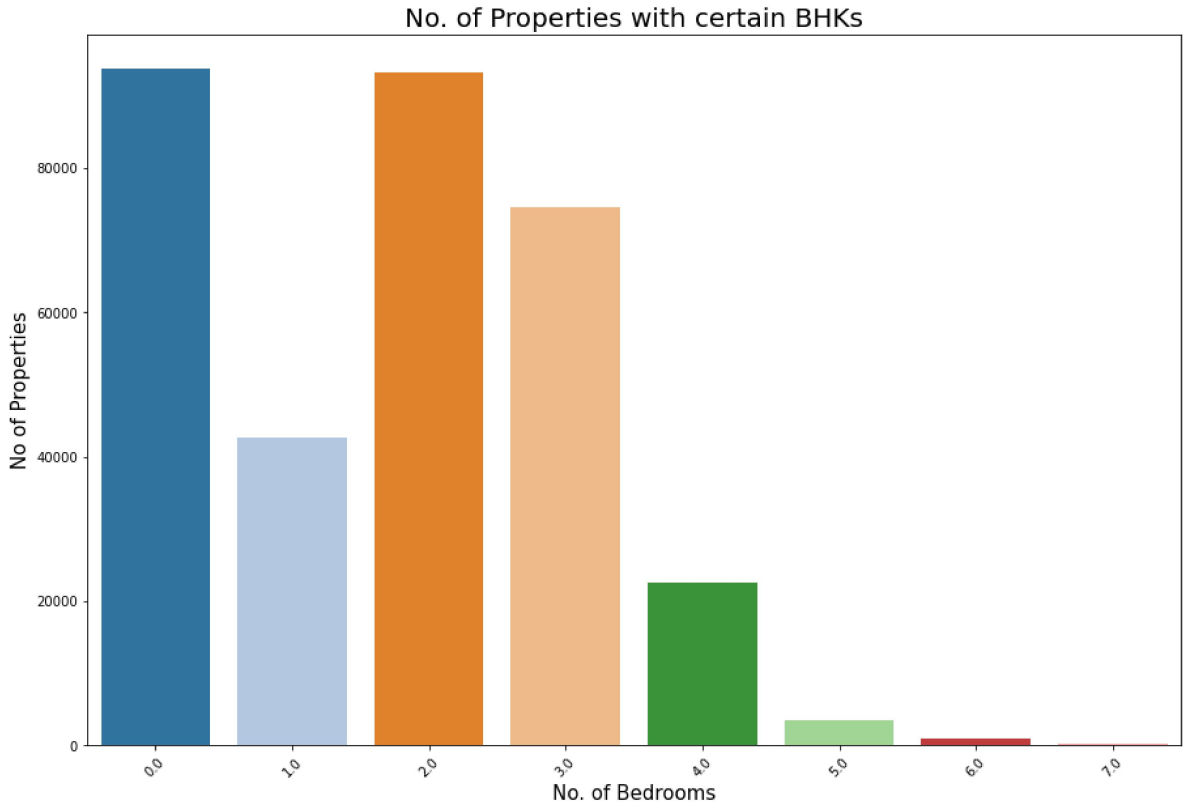
```
plt.figure(figsize = (10, 5))

data = df['is_Apartment'].value_counts()
colors = sns.color_palette('icefire')
labels = ['True', 'False']

plt.pie(data, labels = labels, autopct = '%.0f%%', colors = colors)
plt.title('Total percentage of apartments')

plt.tight_layout()
```



I have showed no.of properties are there in each BHK.

In [29]:

```
fig, ax = plt.subplots(figsize=(15, 10))
ax.set_title('No. of Properties with certain BHKs', fontsize=20)
sns.countplot(x='No_of_BHK', data=df, palette='tab20' )
ax.set_ylabel('No of Properties', fontsize=15)
ax.set_xlabel('No. of Bedrooms', fontsize=15)
plt.xticks(rotation=45)
plt.show()
```



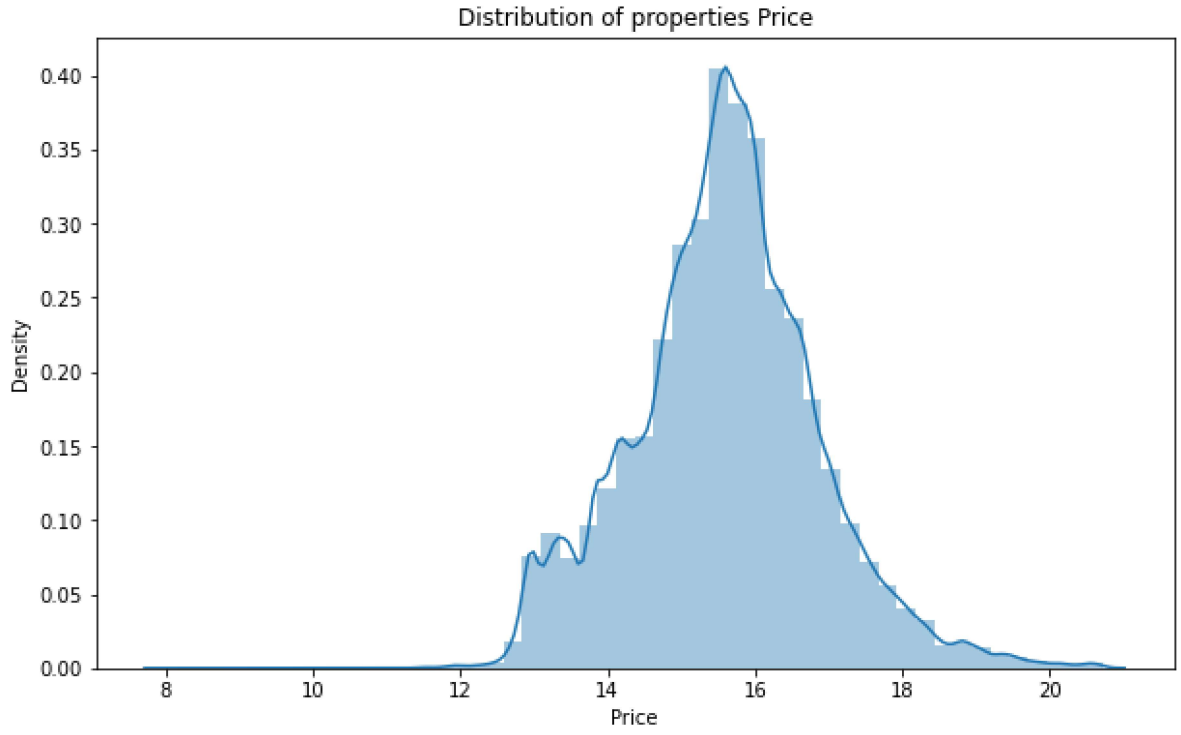
In [30]:

```
px.scatter(df.index, df['listing_domain_score'],
                title="Listing domain score")
```

The distplot represents the univariate distribution of data i.e. data distribution of a variable against the density distribution.Here, i have done distribution of properties price

In [31]:

```
plt.figure(figsize=(10,6))
plt.title("Distribution of properties Price")
dist = sns.distplot(np.log(df['Price']),norm_hist=False)
```



Bivariate Analysis

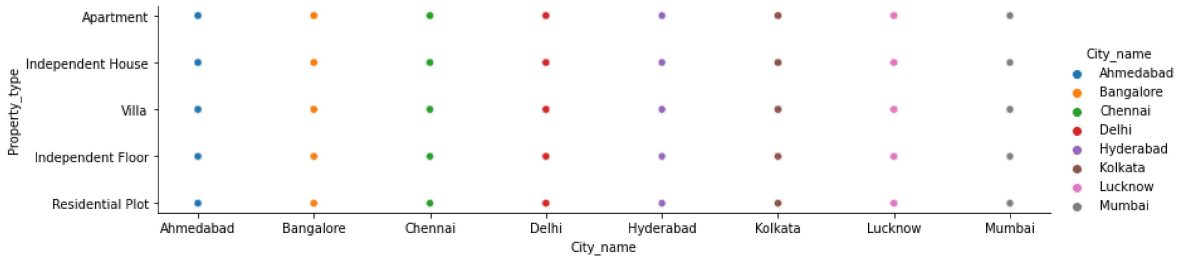
I have used relplot. The one we will use most is relplot(). This is a figure-level function for visualizing statistical relationships using two common approaches: scatter plots and line plots.Here, I have done relplot between city names and Property types.

In [32]:

```
sns.relplot(data=df, x=df["City_name"], y=df["Property_type"], hue=df["City_name"],sizes=(3
```

Out[32]:

```
<seaborn.axisgrid.FacetGrid at 0x1f2b7cd2f40>
```



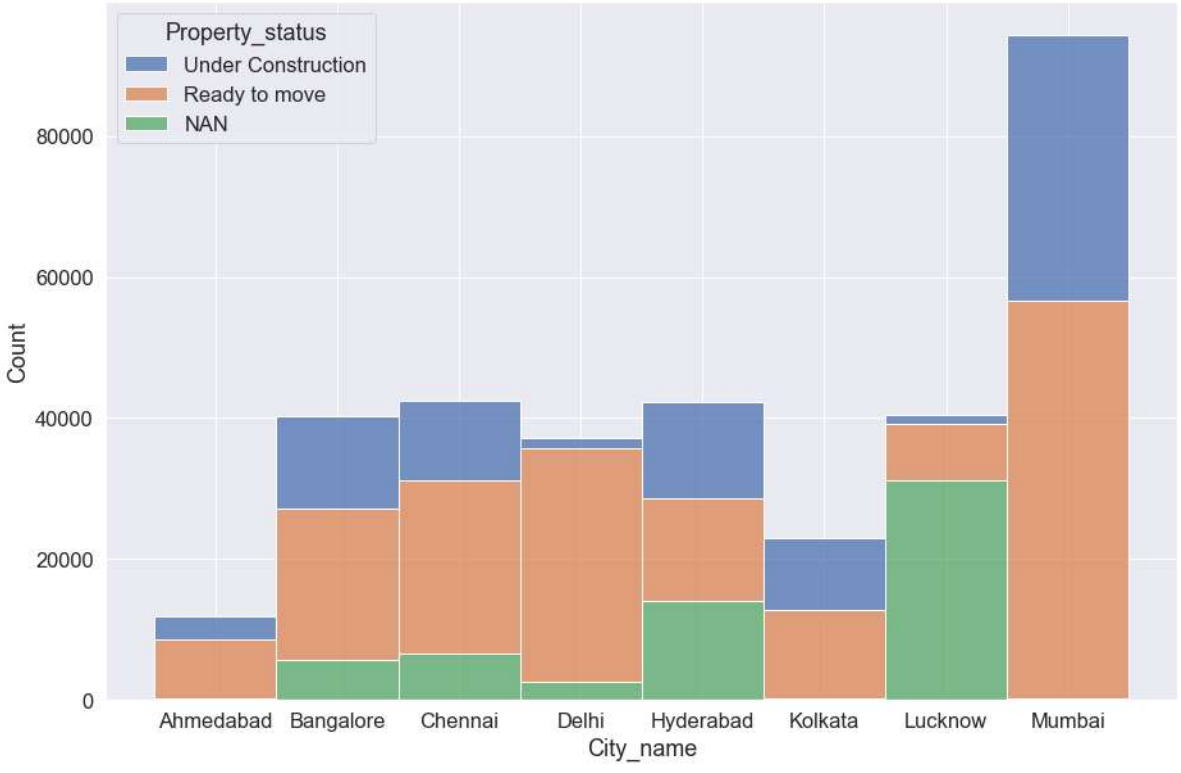
• I have used Histogram to show properties status in each city. It is used to summarize discrete or continuous data that are measured on an interval scale. It is often used to illustrate the major features of the distribution of the data in a convenient form.

In [33]:

```
sns.set(rc={'figure.figsize':(15,10)})
sns.set (font_scale=1.5)
sns.histplot(data=df, x="City_name", hue="Property_status", multiple="stack")
```

Out[33]:

```
<AxesSubplot:xlabel='City_name', ylabel='Count'>
```



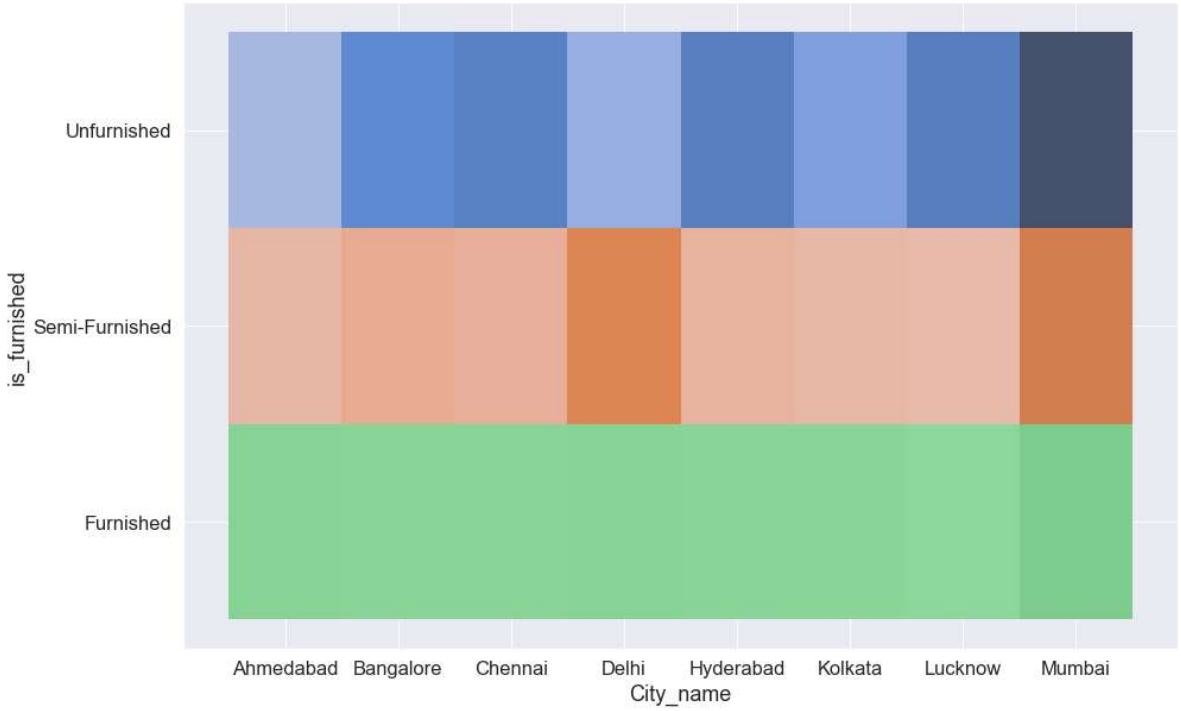
I have used histplot for city names and property is furnished.

In [34]:

```
sns.histplot(df, x="City_name", y="is_furnished", hue="is_furnished", legend=False)
```

Out[34]:

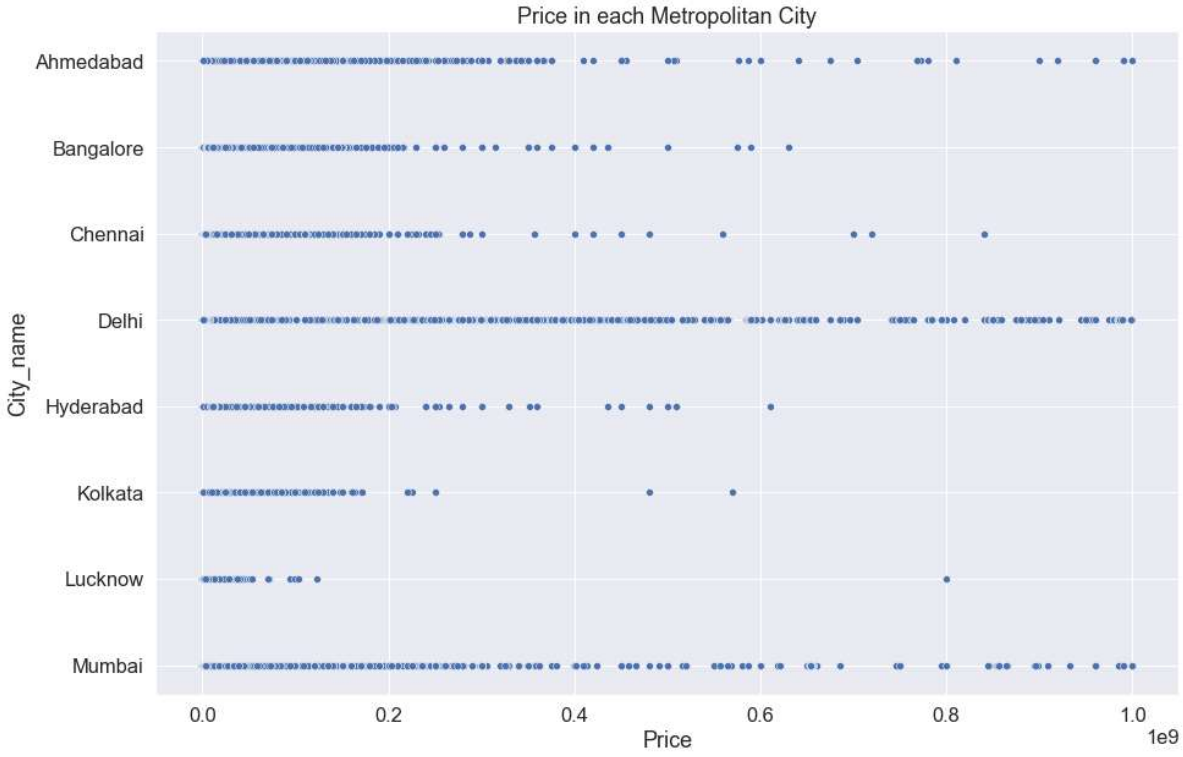
```
<AxesSubplot:xlabel='City_name', ylabel='is_furnished'>
```



• I have done Scatter plot for Numerical and Categorical value. Title is India Metropolitan Areas: Price vs Area

In [35]:

```
sns.scatterplot(x=df['Price'], y=df['City_name'])
plt.xlabel('Price')
plt.ylabel('City_name')
plt.title('Price in each Metropolitan City');
```



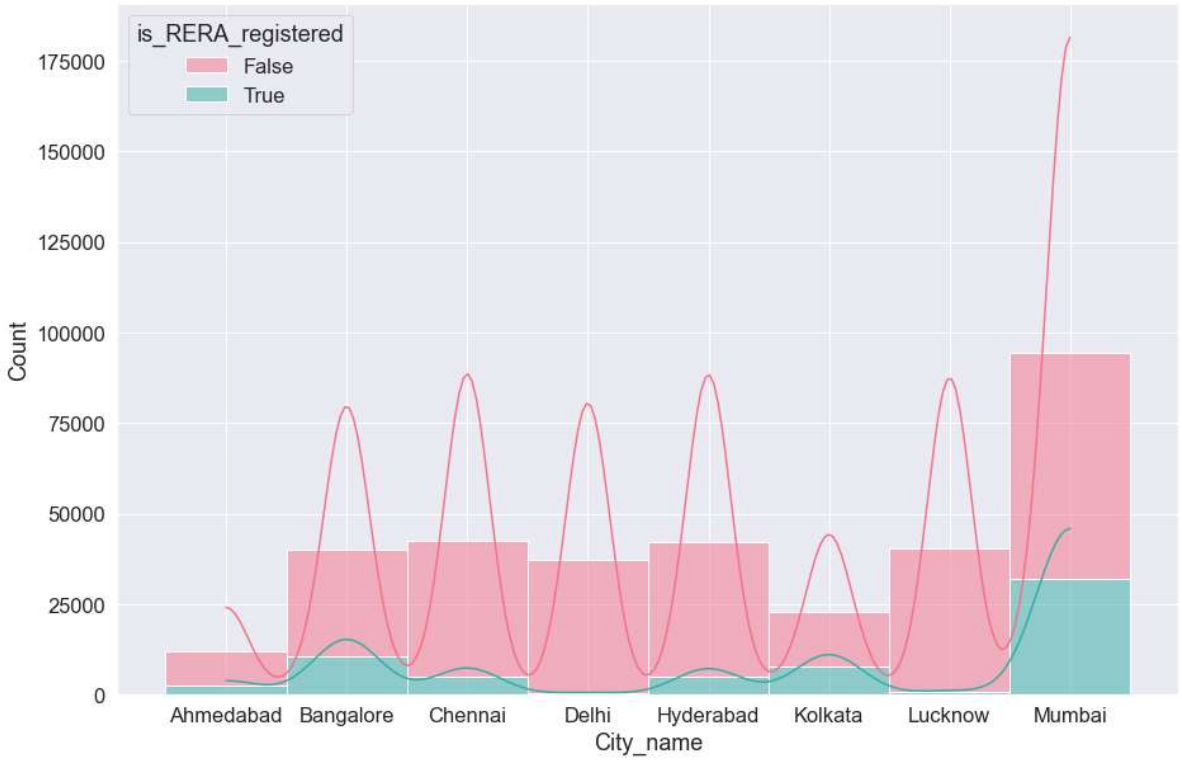
RERA registration is important for gated society. Most properties will be in society. So I have done analysis on city names and RERA registered columns.

In [36]:

```
sns.set(rc={'figure.figsize':(15,10)})
sns.set (font_scale=1.5)
sns.histplot(data=df, x="City_name", hue="is_RERA_registered", multiple="stack",kde=True, p
```

Out[36]:

```
<AxesSubplot:xlabel='City_name', ylabel='Count'>
```



In [37]:

```
df.groupby('No_of_BHK')['is_Apartment'].value_counts().unstack()
```

Out[37]:

is_Apartment	False	True
No_of_BHK		
0.0	93765.0	NaN
1.0	2541.0	40021.0
2.0	13169.0	79943.0
3.0	17413.0	57034.0
4.0	12120.0	10413.0
5.0	2449.0	1147.0
6.0	738.0	238.0
7.0	276.0	38.0

Maping using Folium

• Maping Using Folium: Folium is a powerful Python library that helps you create several types of Leaflet maps. By default, Folium creates a map in a separate HTML file. Since Folium results are interactive, this library is very useful for dashboard building. You can also create inline Jupyter maps in Folium.

• I have imported folium, geopandas, geopy, and I have done plugins HeatMap in it after giving Latitude and Longitude columns Map is shown as below.

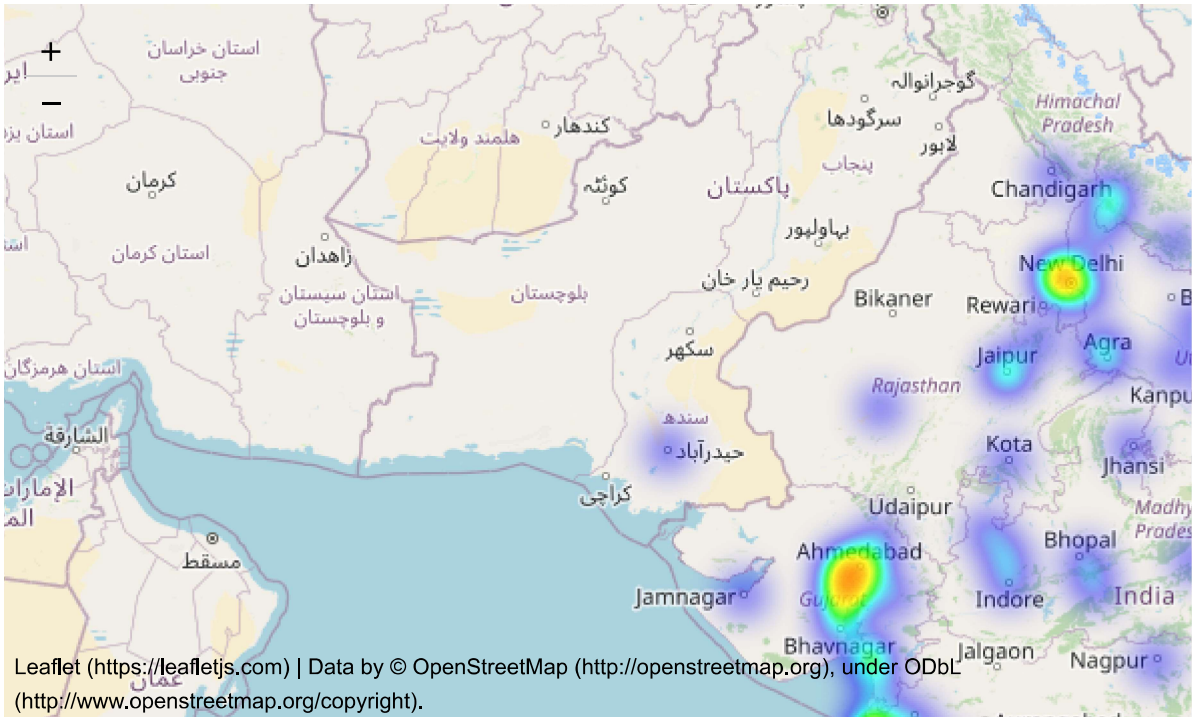
In [38]:

```
import gc
import geopandas as gpd
import matplotlib.pyplot as plt

import folium
from folium.plugins import HeatMap
import geopy as gp
```

In [39]:

```
maps = folium.Map(location = (df['Latitude'].mean(),df['Longitude'].mean()),zoom_start=7,mi
hm = HeatMap(data = df.groupby(['Latitude','Longitude']).size().reset_index(),radius=12,blu
hm.add_to(maps)

maps
```

Out[39]:



Multivariate

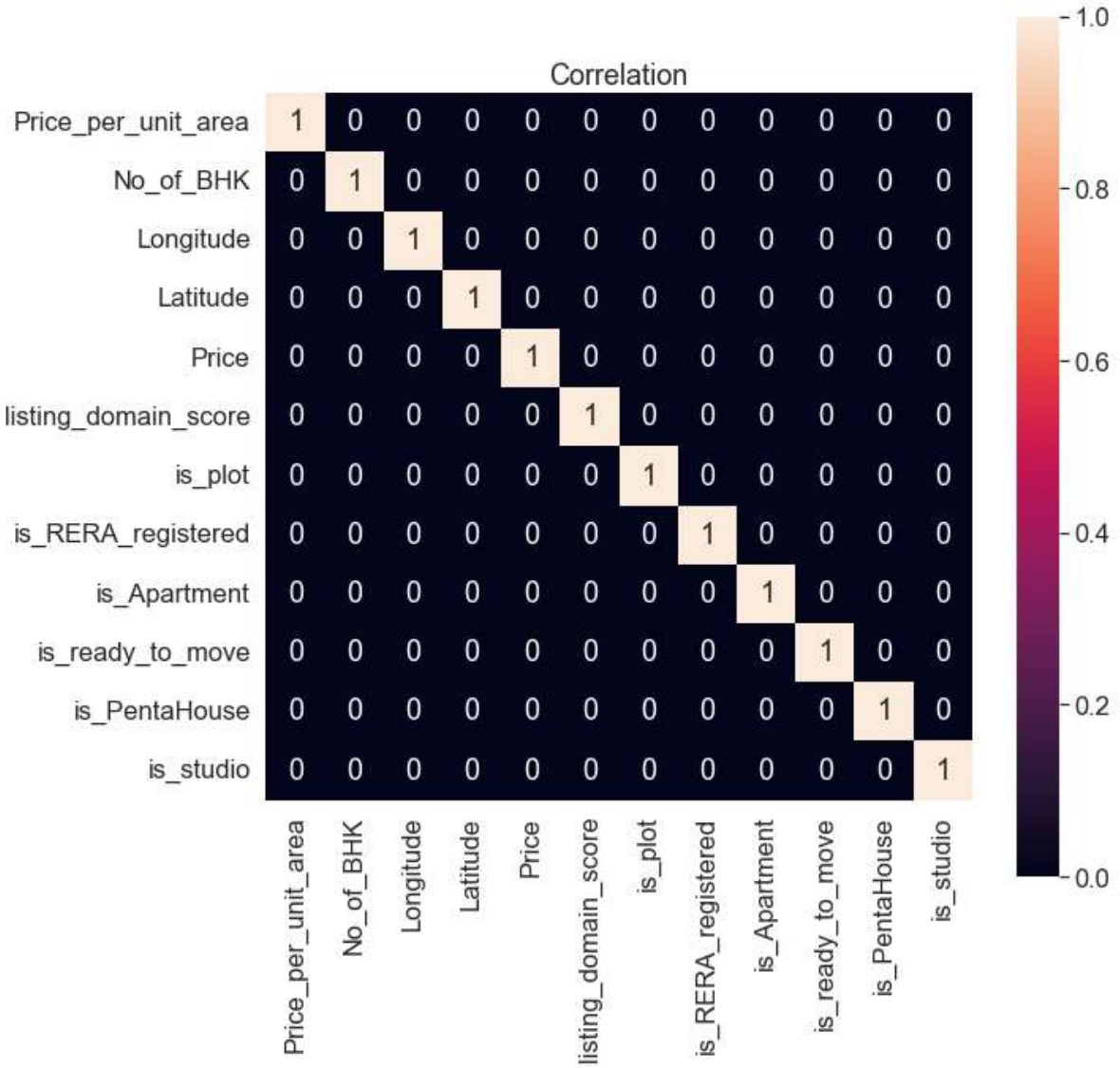
• The statistical study of data where multiple measurements are made on each experimental unit and where the relationships among multivariate measurements and their structure are important. • I have used Heatmaps, a heatmap is a useful visualization method to illustrate multivariate data when there are many variables to compare, such as in a big data analysis. It is a plot that displays values in a color scale in a grid.

In [40]:

```
num=df.select_dtypes(exclude=['object','int'])
numeric_correlation=num.corr()
plt.figure(figsize=(10,10))
plt.title('Correlation')
sns.heatmap(numeric_correlation>0.8, annot=True, square=True)
```

Out[40]:

```
<AxesSubplot:title={'center':'Correlation'}>
```



In [41]:

```
print(numeric_correlation['Price'].sort_values(ascending=False))
```

```
Price                   1.000000
Price_per_unit_area     0.769629
No_of_BHK               0.325955
is_ready_to_move        0.142322
Latitude                0.136961
is_PentaHouse           0.075579
listing_domain_score   -0.005836
is_studio              -0.024560
is_RERA_registered     -0.032716
is_Apartment           -0.057298
Longitude              -0.099958
is_plot                -0.141563
Name: Price, dtype: float64
```

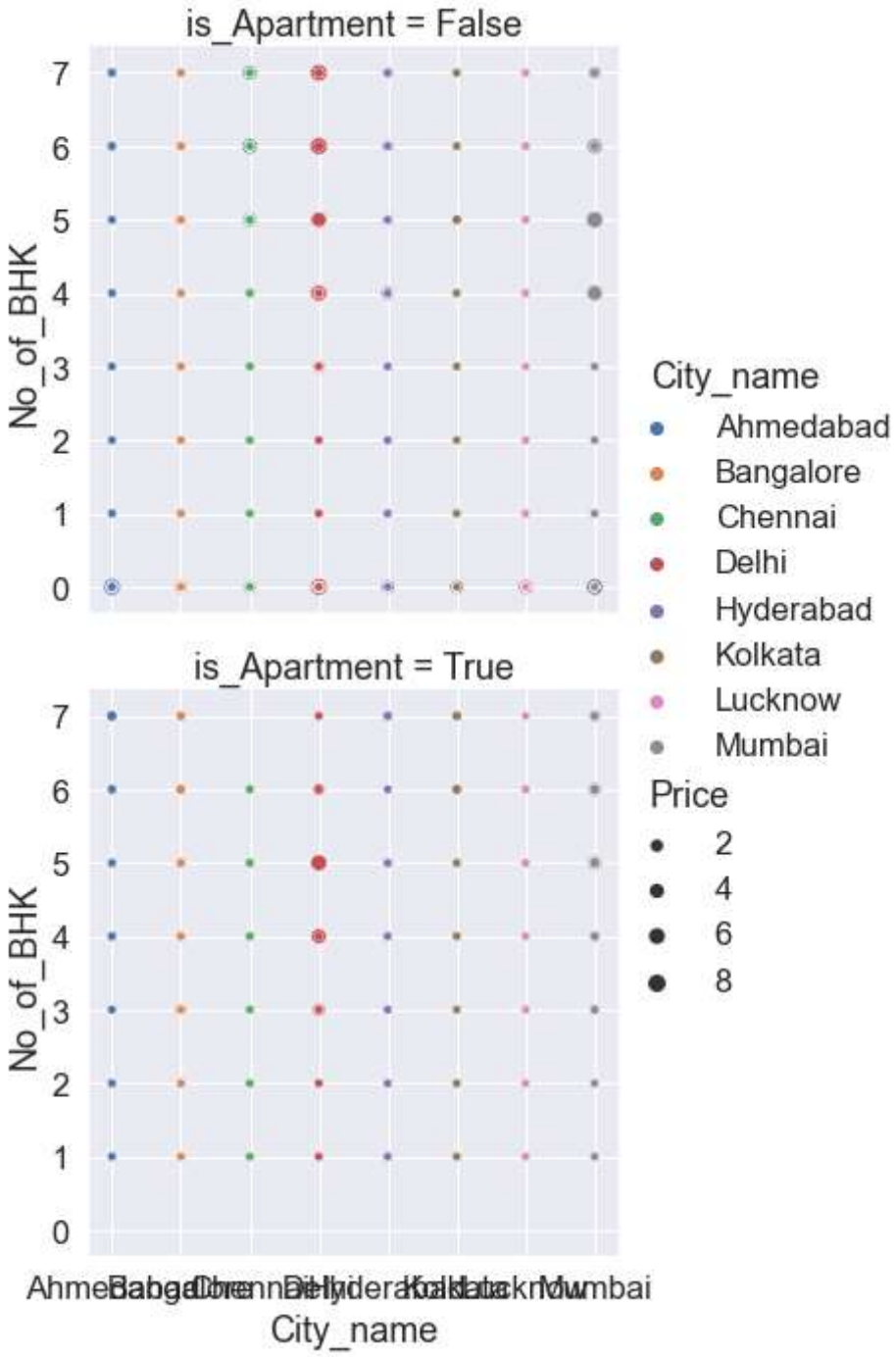
• I have used relpolt Plot for 4 columns, between No_of_BHK, City name, Price and is_Apartment. This plot shows the relationship between two numeric features by using dots to visualize how these variables move together

In [42]:

```
sns.relplot(data=df, x=df["City_name"], y=df["No_of_BHK"], hue=df["City_name"],size="Price"
```

Out[42]:

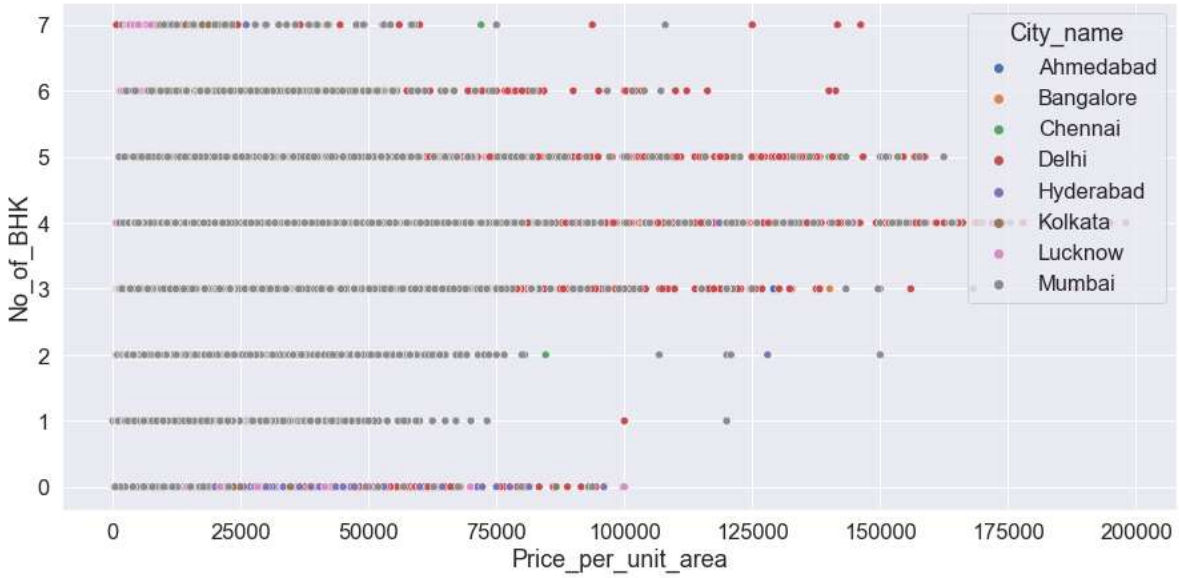
```
<seaborn.axisgrid.FacetGrid at 0x1f2b02935e0>
```



• I have used Scatter Plot for 3 columns, between Price_per_unit_area, No_of_BHK and City name. This plot shows the relationship between two numeric features by using dots to visualize how these variables move together

In [43]:

```
plt.figure(figsize = (15, 7))
sns.scatterplot(x='Price_per_unit_area', y='No_of_BHK', data=df, hue='City_name',palette="d

plt.show()
```



• Maping for Top 30 Locality Name: I have created new dataframe which contains Top 30 Locality Names , Latitude , Longitude. I have used Folium for map and given Locations and Locality names and used marker to mark locations

In [44]:

```
locality_name = df.Locality_Name.value_counts().index[:30]
latitude = df.Latitude.value_counts().index[:30]
longitude = df.Longitude.value_counts().index[:30]
```

In [45]:

```
df2 = pd.DataFrame(locality_name)
df2['Latitude'] = latitude
df2['Longitude'] = longitude
df2.rename(columns = {0:'Locality Name'}, inplace=True)
```

In [46]:

```
a_list = df2[['Locality Name','Latitude','Longitude']].values.tolist()
```

In [47]:

```
property_map= folium.Map(location=[28.71248436,77.15420532])
fg=folium.FeatureGroup(name="Location of Top 25 Locality areas")

for i in a_list:
    fg.add_child(folium.Marker(location=[i[1] ,i[2]],popup=i[0],icon=folium.Icon(color="gre
property_map.add_child(fg)
```

Out[47]:

Conclusion:

• In this paper, we developed several analyses to predict the price of a house based on some of its features. In this paper, we followed the data science process from collecting data to cleaning and preprocessing it, then exploring the data, evaluating the results, and communicating them through visualisations.

• First, we have taken the data set and performed some statistical methods to better understand the dataset. We have checked the shape and got to know that we have 332097 rows, 32 columns of data.

• We should remove the outliers using some statistical methods like considering the values within the 1.5 times the Interquartile Range (which is the statistically ideal range in which outliers are not present). To perform this task, we have calculated the Quartiles. Interquartile Range can be calculated by using the following formula: IQR = Q3 - Q1. The range without outlier is [1.5IQR+Q1, Q3+1.5IQR].

• There are more than 2500 properties that have unit price per square feet more than 3600 Rupees.

• There are 94,272 properties that are under Makaan company in Mumbai which is the highest.

• There are 57% of apartments out of all the properties registered under Makaan.

• There are about 90 thousand houses that are with 2 BHKs and RK houses.

• The distribution of the properties price is a left skewed distribution curve.

• There are more Under Construction Buildings and Ready to move properties in Mumbai.

• There are more empty values in Lucknow.

• There are more Unfurnished, Semi-Furnished and Furnished properties in Mumbai.

• Delhi is more expensive then other cities and Mumbai city have more properties. Some Locality areas are cheap and some are expensive. By this Analysis we can buy properties according to our status.